

# A Privacy Preserving Generative Adversarial Network for Image Data

*Sathish Nallamolu*<sup>1\*</sup> and *Srinivas Padmanabhuni*<sup>2</sup>

<sup>1</sup>M.Tech Student, CS Dept, IIT Tirupati, Yerpedu, 517619, India

<sup>2</sup>Guest Faculty, CS Dept, IIT Tirupati, Yerpedu, 517619, India

**Abstract.** The extensive usage of online applications and social media has raised serious concerns from the public regarding the exposure of their personal information. So, there is a strong need for data anonymization to prevent privacy breaches and leakages. The era of attacks on databases and servers is an old trend. Now, most attacks are based on earning access to users' private data. There are techniques like  $k$ -anonymity and  $l$ -diversity to protect Personally Identifiable Information (PII) from adversaries. However, these techniques still cannot provide security from homogeneity attacks, and their application is limited to structural data only. Till now, the frameworks are only available to anonymize the human face data in image format. In this paper, we proposed a new architecture for protecting privacy-related information in images of Indian vehicle number plates. We propose an architecture for anonymizing the vehicle number plates using Wasserstein's Generative Adversarial Network (WGAN) by retaining the original data distribution even after anonymization. Our framework guarantees that it does not store any information while processing. Our main goal is to protect personal information from the image data. After anonymization, there is no similarity between the original and generated image. Our dataset includes a wide variety of license plates from all regions of India. Our work ensures that no human or a character recognition algorithm can recognize the characters from our anonymized images.

## 1 Introduction

Due to the rapid increase in the usage of social media networks, and the digitalization of data over the past decade, the availability of image data has enormously increased. Human faces, Biometrics, Vehicle number plates, Credit card information, and so on are some cases where the data is exposed to the public with/without the owner's consent. To avoid problems with the use of personal data, General Data Protection Regulation (GDPR), a data privacy regulation, was implemented by the European Union (EU) on 25<sup>th</sup> May 2018. The purpose

---

\* Corresponding author: [cs21m011@iittp.ac.in](mailto:cs21m011@iittp.ac.in)

of GDPR is to protect the personal data of individuals within the EU by imposing strict rules on how data is collected, processed, and stored. The organization has to take the user's permission if they collect any PII data that makes the user identifiable uniquely. The attributes of the data, like Aadhar number, Bank account, Vehicle registration number, and so on, will benefit an adversary in linking to re-identify the data. Recently, the Indian government also introduced the Digital Data Privacy Bill to preserve citizens' privacy.

If the data is in text/tabular format, there are techniques like  $k$ -anonymity [1] [2],  $l$ -diversity, and  $t$ -closeness [3] to preserve privacy. These techniques prevent privacy leakages from attacks like membership inference attacks, model reconstruction attacks, and so on. However, designing an architecture to anonymize the images without any loss of information and retaining the original data distribution is challenging. This problem comes under the generative modelling technique, where new images are formed from the original images. Generative Adversarial Network (GAN) developed by [4] will help with these problems through adversarial training.

The GAN comprises of two neural network architectures: Discriminator and Generator. These two architectures compete with each other like in a minimax game. However, in our architecture, the discriminator acts as a critic, which informs the Wasserstein distance between two data distributions. Our proposed solution successfully anonymizes all vehicle number plate types and creates realistic fake images. We treat the original dataset's images as real and generated images as fake. No human or a character recognition algorithm can read the characters from the anonymized images.

The discriminator network input is an image of resolution  $3 \times 256 \times 256$ . The image is further transformed through a series of convolution layers, and finally, the Wasserstein metric (a scalar value) will be returned. In contrast, the generator consists of transposed convolution layers, which initially take a normal distribution as input and transform it into the original data distribution. Like this, these two networks will compete over the iterations until they converge.

The dataset we considered consists of vehicle number plate images from different parts of India. We chose the dataset to be diverse to encounter all variations of images. The dataset considerably covers a large diversity of brightness, font styles, zooming levels, and so on. We evaluated our performance by taking the percentage of False Negatives in classification. This metric describes the number of fake images identified as real. We used ensembling techniques of machine learning for the evaluation of the model.

In summary, we make the following contributions:

- We proposed a GAN that is privacy preserving, it anonymizes all the vehicle number plates without preserving sensitive information and generates realistic fake images. The generated images contain text which is neither human nor machine readable.
- We selected a diverse dataset containing different images in brightness, zooming levels, font styles, and rotations.

## 2 Related work

Some of the architectures proposed in this domain include Autoencoders, Variational Autoencoders, and GANs. An **Auto Encoder** [5] comprises two neural networks named Encoder and Decoder. A bottleneck layer connects these two networks in the middle. The Encoder is a down sampler i.e., it reduces the dimensionality of the feature vector of the input image and gives it to the bottleneck layer. The Decoder is an upsampler that takes the reduced

feature vector and reconstructs the original image. The bottleneck layer consists of a compressed representation of the input image. The decoder only knows how to upsample the latent vector but does not have any knowledge about the probability distribution of the original data. So, these cannot act as a generative model. To overcome that Variational Autoencoders are developed.

**Variational Autoencoders** [6] are used in generative modeling. The goal of VAE is to find a distribution  $q_{\phi}(z|x)$  of some latent variables ( $z$ ), which are sampled from  $z \sim q_{\phi}(z|x)$  to generate new samples  $x' \sim p_{\theta}(x|z)$  where  $x$  is a datapoint from the original data,  $q_{\phi}$  is an encoder,  $p_{\theta}$  is a decoder and  $\Phi, \theta$  and are weights of the encoder and decoder respectively. This means VAE can learn about the original probability distribution. Autoencoder encodes a single value about a feature (latent variable), whereas the VAE encodes the feature into a probability distribution, i.e., a range of values. Each feature is sampled from its distribution and given to the decoder. So, Variational autoencoders act as a generative model. In the loss function, the first term is data (image) reconstruction loss, and the second term is a regularization term that measures the similarity between both distributions using the *Kullback–Leibler (KL)-divergence* test [7].

$$L(\phi, \theta) = \mathbb{E}_{q_{\phi}(z|x)}\{p_{\theta}(x|z)\} + KL\{q_{\phi}(z|x) || p_{\theta}(z)\} \quad (1)$$

Where the KL-divergence is defined as:  $KL(P || Q) = \sum p(x) \log(p(x)/q(x))$  (2)

Some things could be improved in the KL divergence metric. It returns  $\infty$  even if both distributions are almost similar. *KL-divergence* test is not symmetric. This implies that  $KL(P || Q)$  and  $KL(Q || P)$  return different results. So this generative modelling cannot generate better fakes than the GANs do. To avoid this problem, GANs follow the Jensen–Shannon (JS) divergence metric to minimize the distance between the probability distributions.

**GANs** are one of the very fundamental mechanisms used for generative modeling. A GAN consists of two neural network architectures named discriminator and generator. The discriminator classifies whether the input image is real or fake. The real images come from the dataset chosen. The fake images come from the generator. The generator generates new images by approximating the original data distribution, starting with a normal distribution. Theoretically, the generator's goal is to fool the discriminator by generating fake images that look real [8]. These two networks compete with each other like in minimax game theory. The discriminator wants to maximize its prediction capability, whereas the generator wants the exact opposite of it. This process is called as Nash Equilibrium by [9]. There are hundreds of GAN architectures available today. There is at least one GAN available per alphabet. The applications of GANs are not limited to images. There are GANs for text generation, video processing, medical data preservation, noise removal in multimedia applications, image segmentation, and so on. In our work, we show how a GAN perfectly suits the task of privacy preservation in images. The binary cross-entropy loss can be used as a loss function. The loss function of GAN can be defined as:

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim P_{data}^x} \{\ln(D(x))\} + \mathbb{E}_{z \sim P_z} \{\ln(1 - D(G(z)))\} \quad (3)$$

We try to maximize the classification performance of the discriminator. Based on the classification, the generator updates the noise and transforms it similarly to the original image. GAN follows the Jensen-Shannon divergence to measure the difference between the probability distributions. JS divergence measures how one probability distribution is different

from another. For two discrete probability distributions,  $P$  and  $Q$ , the JS divergence [10] between them is defined as

$$JS(P || Q) = \{ KL(P || M) + KL(Q || M) \} \quad (4)$$

Where,  $M$  is the average probability distribution of  $P$  and  $Q$ , i.e.,  $M = (P+Q)/2$ . The result of JS-divergence ranges from 0 to  $\log 2$ .

The main problem with the JS-divergence test is the calculation of new distribution  $M$  which is computationally heavy. Although JS divergence is often used as a distance metric between probability distributions, it does not satisfy all the properties of an accurate distance metric, such as triangle inequality. Therefore, it may only sometimes provide a meaningful measure of similarity between distributions.

Image inpainting techniques [11] [12] help fill the missing pixels of the image. This technique replaces/fills the pixels by observing the features from the neighborhood. This technique of replacing the pixels can be applied to the regions of the image where the sensitive information is there. However, these inpainting techniques will only work well in some cases. Sometimes, they cannot remove the privacy-sensitive information. The attackers will try to get the information by comparing the neighborhood areas and returning the original image. Sometimes the pixels will be replaced just with black or white pixels. This approach may spoil our ultimate goal of retaining original data distribution. These image inpainting techniques work primarily for inpainting human faces. The techniques generate landmark points from the faces like eyes, nose, mouth, and so on. This is done by extracting the features from a wide variety of faces. New images are generated by modifying these landmarks by adding some noise. This method works well when the faces are visible in the image. This model is slightly complicated if the faces are blurred or blacked out. This technique will only work for a limited variety of faces and poses. This architecture cannot handle diverse datasets. Since this approach has disadvantages, we moved on to use GANs, which can learn the overall probability distribution.

The main goal of our problem is to remove all the sensitive information from images. Simply applying a blur filter will not work, as convolution can be used. Convolution operation works by replacing the current pixel value with the sum of products of pixels in the image and filter. Since convolution operation is a combination of arithmetic operations, a deblurring filter can be designed, and the original image can be reconstructed. We are looking for a guarantee that no one should be able to reconstruct the original image.

The  $k$ -same algorithm [13] implements the  $k$ -anonymity algorithm for face images. The  $k$ -same algorithm describes that in a set of  $k$  images, no two images should have the same distribution of pixels in sensitive information regions. It has been proved that the  $k$ -same algorithm can remove all privacy-sensitive information, but the resulting images will form ghosting images. In ghost images, the human face will not look like a face. As a result, the original and fake images look dissimilar. We may not achieve the  $k$ -same property in our problem because all the license plates only have English and numerical characters. Nevertheless, our algorithm did not result in any ghosting images.

### 3 Proposed work

In the case of the face identification, features like eyes, nose, gender, mouth, and so on matter most. However, in number plates, there are no such features. The ultimate goal is to make the number plate characters unreadable. So a standard GAN architecture will not work in our

case. The GAN should be more powerful and able to minimize the distance between the original and latent distribution. We tried Deep Convolutional GAN (DCGAN) [14] and Conditional GAN (CGAN) [15] before working out with WGAN. The results are not satisfactory, and the resulting images are more of random black pixels. This result indicates that these GANs cannot correctly learn or approximate the original data distribution. So the GAN we chose for our problem should be able to learn or approximate the original data distribution and generate the fake images. The loss function is the main problem behind their poor performance in DCGAN and CGAN. The loss function should differ from the traditional GAN architectures for our problem.

WGAN [16] is a type of GAN that uses a different loss function than the traditional GAN. Rather than using the binary cross-entropy loss function used by traditional GANs, the WGAN uses a metric named Wasserstein distance, also known as the Earth Mover's Distance (EMD), as its loss function. For measuring the distance between two distributions, the other divergence techniques use vertical distance (taking the probability scores), whereas the WGAN uses the horizontal distance between the distributions. The Wasserstein distance measures the minimum amount of work required to transform one probability distribution into another. In the context of GANs, the Wasserstein metric measures the distance between the generated and actual samples' distribution. By minimizing the Wasserstein distance between these two distributions, the generator learns to produce samples that are more similar to the actual samples. Compared to other GANs, the training of WGAN is more stable.

The loss function of WGAN is defined as:

$$\min G \max D = \mathbb{E}_{x \sim P_{data}} \{D(x)\} - \mathbb{E}_{z \sim P_z} \{D(G(z))\} \quad (5)$$

The loss function returns the distance between the original distribution and the latent distribution. This helps the generator update the weights according to the distance returned. If the distance is high, the generator has to generate much better fakes. If the distance is low, the generator is able to generate samples similar to the original data and about to converge.

### 3.1 Generator Architecture:

Our proposed generator architecture consists of three parts: transposed convolution layers, batch normalization, and activation functions. Transposed convolution layers act as an upsampler for generating the images from a latent vector of features. This is often called as random noise vector. This latent vector is similar to the bottleneck layer in autoencoders, except that it does not know about the compressed features of original data. The images are generated over the iterations by molding the initial distribution (normal distribution) to fit over the original distribution.

After each layer, batch normalization is applied to every transposed convolution layer. This technique is used to normalize the activations of a neural network, reducing the internal covariate shift between layers, which can accelerate the training process and improve generalization performance.

For all the layers except the output layer, the *ReLU* activation function has been used. *Tanh* is used as the activation function for the output layer. The range of the *tanh* function is [-1, 1]. This range is useful for generating images or data that have pixel values in the same range. Moreover, the *tanh* function is differentiable, which is necessary for backpropagation during the training of the GAN.

Our generator takes the latent vector as input and outputs the image of size 256×256, which is the size of our original image. The generator progressively grows to 4×4, later 8×8

until the size of the original image. Later, these images are fed to the discriminator for classification. Fig.1 describes the architecture of our generator.

### 3.2 Discriminator Architecture:

Our proposed discriminator architecture also consists of three parts: convolution layers, instance normalization, and activation functions. The discriminator is basically a binary classifier network. It downsamples the input image by convolving it with multiple kernels of different dimensions. The downsampling of the image continues until the output ranges from  $[0, 1]$ . However, in our case, the discriminator is acting like a **critic** instead of a classifier. This is keeping our GAN on top. The discriminator returns the horizontal distance (a real number) between the original and latent distribution. This value helps the generator understand its performance and update the weights accordingly. InstanceNorm2d normalizes the inputs of a 2D convolutional layer across the channel dimension (i.e., along the depth dimension). Unlike batch normalization, which normalizes the inputs across the batch dimension, InstanceNorm2d normalizes the inputs for each individual image separately. By normalizing the inputs in this way, InstanceNorm2d can help reduce internal covariate shifts and improve the stability and convergence of the neural network during training. *LeakyReLU* is used as an activation function for the convolution layers with  $\alpha = 0.2$ . The objective of the discriminator is to maximize the difference between the mean scores of real and fake samples, which is equivalent to minimizing the Wasserstein distance between the two distributions. For each training of the generator, we are training our discriminator five times. This is to make the discriminator strong, and as a result, the generator will output diverse fakes. Otherwise, mode collapse problem will occur. Fig.1 describes the architecture of our discriminator. We will discuss mode collapse later in this paper. During the training, the weights of both the discriminator and generator are updated using the gradient of the Wasserstein distance with respect to their parameters. The gradient penalty ensures that the discriminator's gradient is Lipschitz continuous, which is a necessary condition for the Wasserstein distance to be well-defined. The Lipschitz continuity is a function property that limits how fast the function can change.

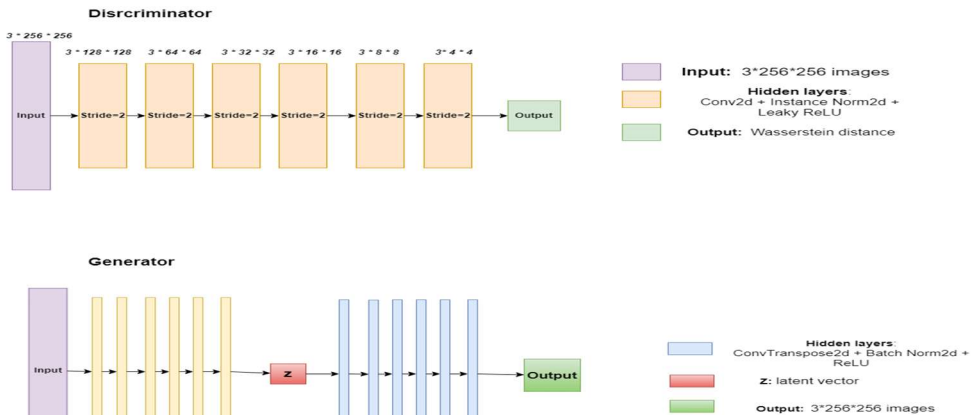


Fig. 1. Architecture of Discriminator and Generator

### 3.3 Lipschitz continuity

The Lipschitz continuity for the weights of the discriminator in WGAN can be achieved in two ways. One is by setting a clip value and then restricting the weights of the discriminator to be within the range of the clip value. However, weight clipping can cause several issues, such as making it difficult to optimize the model and leading to mode collapse. Mode collapse is a scenario when the generator fails to approximate the original distribution and generates fewer quality images or similar output all the time.

The main reason for mode collapse is a weak discriminator. In the initial phases of training, the discriminator will be accurate and able to classify all the images correctly. As a result, the weights of the discriminator will not be updated, and no training will happen. However, we want the discriminator as a strong opponent to the generator. So we trained our discriminator five times per each training of the generator. This makes the discriminator so powerful, and the generator can also output better fakes. Our algorithm used the gradient penalty in the loss function as a regularization term. Gradient penalty avoids the situation of mode collapse [17] and vanishing gradients [18].

An alternative method to enforce Lipschitz continuity is to use a gradient penalty, which involves adding a term to the loss function that penalizes the norm of the discriminator's gradient concerning its inputs. This method is more effective than weight clipping in stabilizing WGANs and preventing mode collapse. We used this in our architecture.

### 3.4 Gradient Penalty:

The gradient penalty term penalizes the norm of the gradient of the discriminator output concerning its input when evaluated at random points along the straight line between pairs of real and generated samples. This encourages the discriminator to have a Lipschitz constant of 1, which ensures that the Wasserstein distance is well-defined and differentiable almost everywhere. By adding the gradient penalty term to the WGAN loss function, we can improve the stability of the training process and prevent the discriminator from collapsing to a constant function. The gradient penalty term can be expressed as follows:

$$GP = \lambda * ||\nabla_x D(\bar{X}) - 1 ||^2 \tag{6}$$

Where  $\lambda$  is a hyperparameter that controls the strength of the penalty term,  $\bar{X}$  is a random point on the straight line between a real and a generated sample, and  $D$  is the discriminator. The total loss of the WGAN with gradient penalty is then:  $L = D(x) - D(G(z)) + GP$  where  $x$  is a real sample,  $G$  is the generator,  $z$  is a noise vector, and  $D$  is the discriminator. The loss function measures the difference in the distance between real and fake samples.

The best thing about our model is that we generate multiple fake images for a single real image. This prevents adversarial attackers from mapping the real and fake images. The chances of mapping the exact real and the corresponding fake are almost impossible with any kind of adversarial attack. This feature is not available in any GANs developed until now. This is the reason behind the success of our GAN.

## 4 Experiments

For our experiment, we used the Indian Number Plates Dataset from Kaggle. This dataset is primarily used for detection, classification, and optical character recognition tasks. The main reason behind choosing this dataset only because this is the only dataset having images with characters. This demo dataset consists of around 1200 realistic images of vehicle number plates captured from various regions of India. The dataset consists of a vast diversity of font styles, different shapes of number plates, contrast levels, colors, brightness levels, orientations, and so on.

Our final model is trained on a Kaggle kernel with 12 GB RAM and Nvidia P50 GPU and took around 7 hours for 150 epochs of training. We can generate multiple fakes for a single image, making our model more reliable than others. This prevents the attacks like membership inference attacks, model inversion attacks, and so on.

The results of our experiments are shown in Fig. 2 and Fig. 3. Fig. 2 contains the original images used for training the model, and Fig. 3 contains the fake images which are generated by our model.



**Fig. 2.** Sample images from the Indian vehicle number plates dataset



**Fig. 3.** Anonymized images generated by our GAN. The characters appearing in the images are not readable or recognizable.

## 4.1 Privacy Focused Adversarial Attacks on Machine Learning Models

Some of the major attacks on databases for privacy leakages are *membership inference attacks*, *property inference attacks*, and *model extraction attacks* [19]. Depending upon the architecture of the model, the inference attack is chosen.

**Membership inference attack:** The adversary may not know the model's internal parameters. However, the attacker knows the model's architecture, i.e., SVM, Neural networks, KNN, GAN, or the service used by the model. These attacks take advantage of discovering or reconstructing the examples used to train the model. So this leads to the privacy leakage of the users whose data was used during the training of the model. Here the attackers know the service (algorithm/architecture) that the developer used so that they can use the same service to create a similar model.

The attacking technique is as follows: For a target model that is stored in the cloud, the attacker generates random records. The attacker supplies the model with some random data. Based on the confidence score the model gives for that input, the attacker modifies the features of the input and reruns it by the same model. The same process is repeated until the model reaches a high confidence score. The record is similar to one of the data records used while training the model. After gathering enough high-confidence records, the attacker creates multiple models for training the data. These models are called shadow models. All these shadow models together form a membership inference model. This final model can predict whether a record was included in the training dataset of the target machine-learning model. These membership inference attacks will be able to gather sensitive information from tabular data. However, these will not work correctly if the dataset contains high-resolution images and multi-class classification because it needs to explore a large feature space. So our model will successfully defend against this type of attack.

**Model inversion attack:** This attack is also similar to the membership inference attack, where the attacker tries to find whether a particular record has been used in the training data or not. Nevertheless, this attack is mostly made on deep-learning models. ML model is just a composition of functions. It will use the parameters of the function to predict the output of test data. These parameters are obtained from the training data. The attacker tries to reconstruct unknown inputs based on just model outputs. The attacker gives the input that is used in training as similar as possible.

Suppose the attacker wants to retrieve the training data from a hand-written characters recognition dataset. So the attacker will give the characters from multiple languages as input to the model and observes the confidence score of each character. By repeating the task several times, the attacker can come to a conclusion about the language used in training. After knowing about the training data, the attacker tries to train the model with his data. This is a severe problem. There are similar attacks named property inference attacks and model extraction attacks. Depending upon the architecture of the model, the inference attack is chosen.

Since the characters in the fake images are not recognized by any recognition algorithm, our model is more robust to attacks. Even if the attacker is from within the organization, the attacker will have access to anonymized images only. So no attacks can gain access to the original data.

## 4.2 Analysis of the fake images

We developed an ensemble algorithm (combination of multiple ML models) to identify the fake images which are being identified as real.

We built two binary classification models for classifying real and fake images. The first model is trained on real images, and the second is trained on fake images. Later the ensemble model is built by combining both models. The main reason for building this analysis algorithm is to identify the number of False Negatives from the fake images. False Negatives represent the number of originally fake images; however, the model classified them as real. The result of our algorithm returned zero false negatives. It represents that our fake images are not being identified as real. From this, we can conclude that our model generates fake images not identified as real in adversarial attacks.

The ensemble algorithm is acting like a judge for our WGAN. This algorithm evaluates the performance of the GAN. In the future, we plan to build algorithms using all types of ensembling techniques and validate the performance of our GAN architecture.

## 5 Conclusion

In this paper, we proposed a privacy preserving architecture for anonymizing the vehicle number plates using WGAN by keeping the original data distribution undisturbed. We designed a generator that generates fake images by approximating the original distribution starting from the latent distribution of features. Our model generated fake images such that humans or any language recognition algorithms do not recognize the characters in those images. After getting the output, the images are analyzed by building an ensemble algorithm for the number of false negatives. There are zero false negatives. This makes our algorithm more reliable and robust for anonymizing the images where text is involved. Moreover, our algorithm can defend the adversarial attacks. The attacker can only gain access to fake images but not real ones. The original data is always safe so that no privacy leakages will happen from the model's side. We believe our contribution will lead to further research into privacy preservation in machine learning.

## References

1. L. Sweeney, "Achieving K-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, p. 571–588, 2002.
2. L. Sweeney, "k-anonymity: A model for protecting privacy," *International journal of uncertainty, fuzziness and knowledge-based systems*, vol. 10, p. 557–570, 2002.
3. N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *2007 IEEE 23rd international conference on data engineering*, 2006.
4. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, p. 139–144, 2020.
5. Y. Yang, Q. J. Wu and Y. Wang, "Autoencoder with invertible functions for dimension reduction and image reconstruction," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, p. 1065–1079, 2016.

6. D. P. Kingma, M. Welling and others, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, p. 307–392, 2019.
7. F. Pérez-Cruz, "Kullback-Leibler divergence estimation of continuous distributions," in *2008 IEEE international symposium on information theory*, 2008.
8. I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.
9. H. K. Khanuja and A. A. Agarkar, "Towards GAN Challenges and Its Optimal Solutions," *Generative Adversarial Networks and Deep Learning: Theory and Applications*, 2023.
10. M. a. P. J. a. P. L. a. P. M. Menendez, "The Jensen-Shannon divergence," *Journal of the Franklin Institute*, vol. 334, pp. 307--318, 1997.
11. Q. Sun, L. Ma, S. J. Oh, L. Van Gool, B. Schiele and M. Fritz, "Natural and effective obfuscation by head inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
12. H. Hukkelås, F. Lindseth and R. Mester, "Image inpainting with learnable feature imputation," in *DAGM German Conference on Pattern Recognition*, 2020.
13. R. a. S. L. a. D. I. T. F. a. B. S. Gross, "Model-based face de-identification," in *2006 Conference on computer vision and pattern recognition workshop (CVPRW'06)*, 2006.
14. A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.
15. H. Hukkelås, R. Mester and F. Lindseth, "Deepprivacy: A generative adversarial network for face anonymization," in *International Symposium on visual computing*, 2019.
16. M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, 2017.
17. K. Zhang, "On mode collapse in generative adversarial networks," in *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II* 30, 2021.
18. Z. Ding, S. Jiang, and J. Zhao, "Take a close look at mode collapse and vanishing gradient in GAN," in *2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)*, 2022.
19. M. Al-Rubaie and J. M. Chang, "Privacy-Preserving Machine Learning: Threats and Solutions," *IEEE Security & Privacy*, vol. 17, pp. 49-58, 2019.